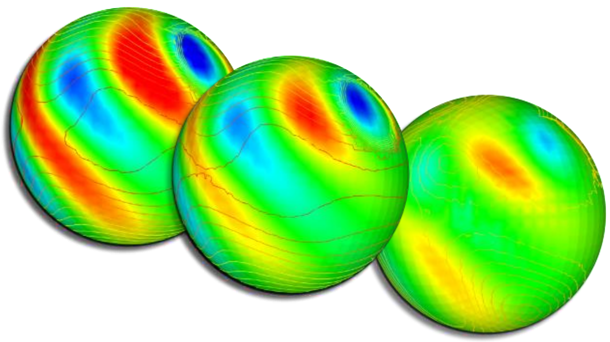
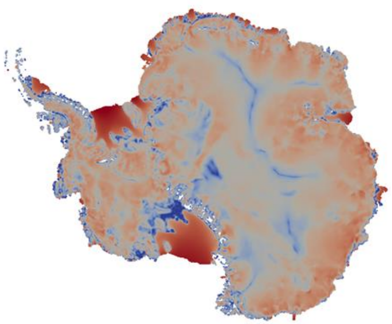
Exceptional service in the national interest

Sandia National Laboratories





Trilinos

Albany

A Performance-Portable Implementation of the Finite Element Assembly in an Atmosphere & Land-Ice Code using the Kokkos Library

Irina Tezaur[1], Jerry Watkins[1,2], Irina Demeshko[3]

[1] Sandia National Laboratories, [2] Stanford University, [3] Los Alamos National Laboratory

FEF 2017 Rome, Italy April 5-7, 2017

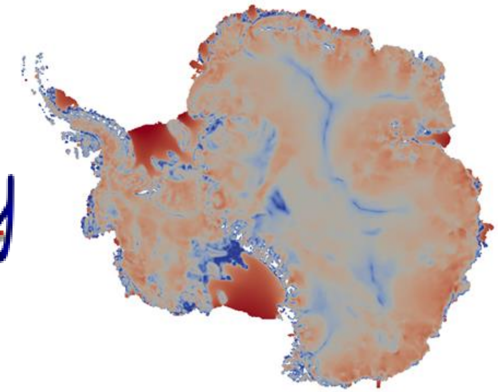
U.S. DEPARTMENT OF ENERGY

NNSA National Nuclear Security Administration

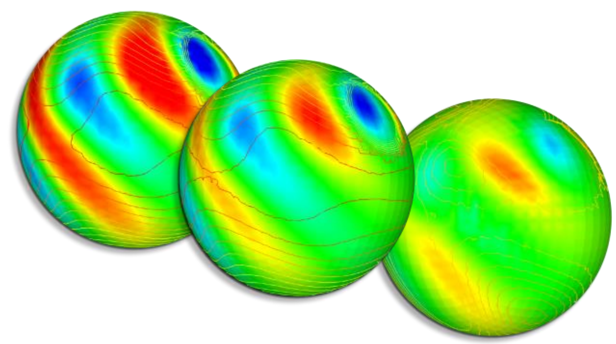
Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000. SAND NO. 2011-XXXXP

SAND2017-2917C

Outline



- Motivation & Background
- The Albany Multi-Physics Code
- Performance-portability via Kokkos
- Aeras Next-Generation Global Atmosphere Model & Project
  - Results
- FELIX Land-Ice Model & PISCEES Project
  - Results
- Summary/Conclusions
- Ongoing/Future Work

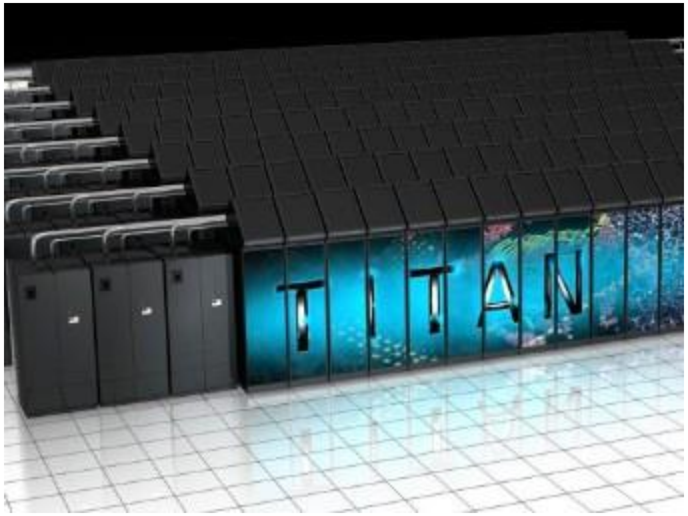


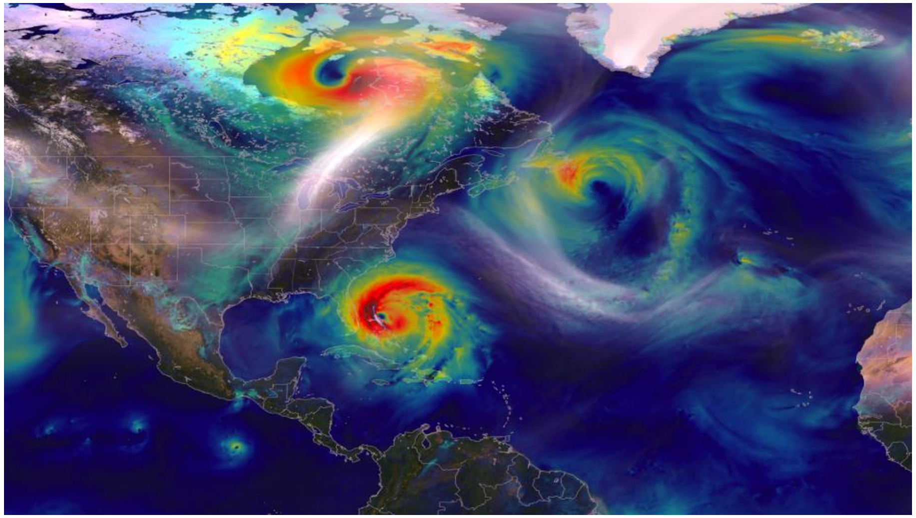
Motivation

- Scientific models (e.g. climate models) need more ***computational power*** to achieve ***higher resolutions***.

- High performance computing (HPC) architectures are becoming increasingly more ***heterogeneous*** in a move towards ***exascale***.

- Climate models need to adapt to execute ***correctly*** & ***efficiently*** on new HPC architectures with drastically ***different memory models***.





MPI+X Programming Model

Sandia National Laboratories

- HPC architectures are rapidly changing, but ***trends*** remain the same.
  - ***Computations*** are ***cheap***, ***memory transfer*** is ***expensive***.
  - ***Single core cycle*** time has improved but stagnated.
  - Increased ***computational power*** achieved through ***manycore architectures***.

  → MPI-only is not enough to exploit emerging massively parallel architectures.

Year	Memory Access Time	Single Core Cycle Time
1980s	~100 ns	~100 ns
Today	~50-100 ns	~1 ns

Approach: MPI+X Programming Model

- **MPI:** inter-node parallelism.
- **X:** intra-node parallelism.

  → *Examples:* X = OpenMP, CUDA, Pthreads, etc.

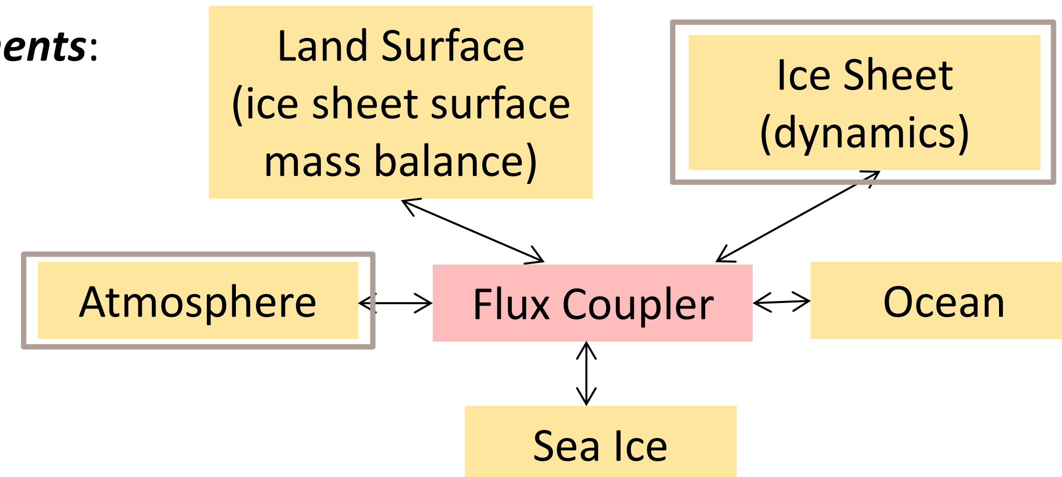
Earth System Models: CESM, DOE-ESM

NCAR
NATIONAL CENTER FOR ATMOSPHERIC RESEARCH

ACME
Accelerated Climate Model for Energy

Sandia National Laboratories

- An ESM has ***six modular components***:
  1. Atmosphere model
  2. Ocean model
  3. Sea ice model
  4. Land ice model
  5. Land model
  6. Flux coupler



Goal of ESM: to provide actionable scientific predictions of 21st century sea-level rise (including uncertainty).

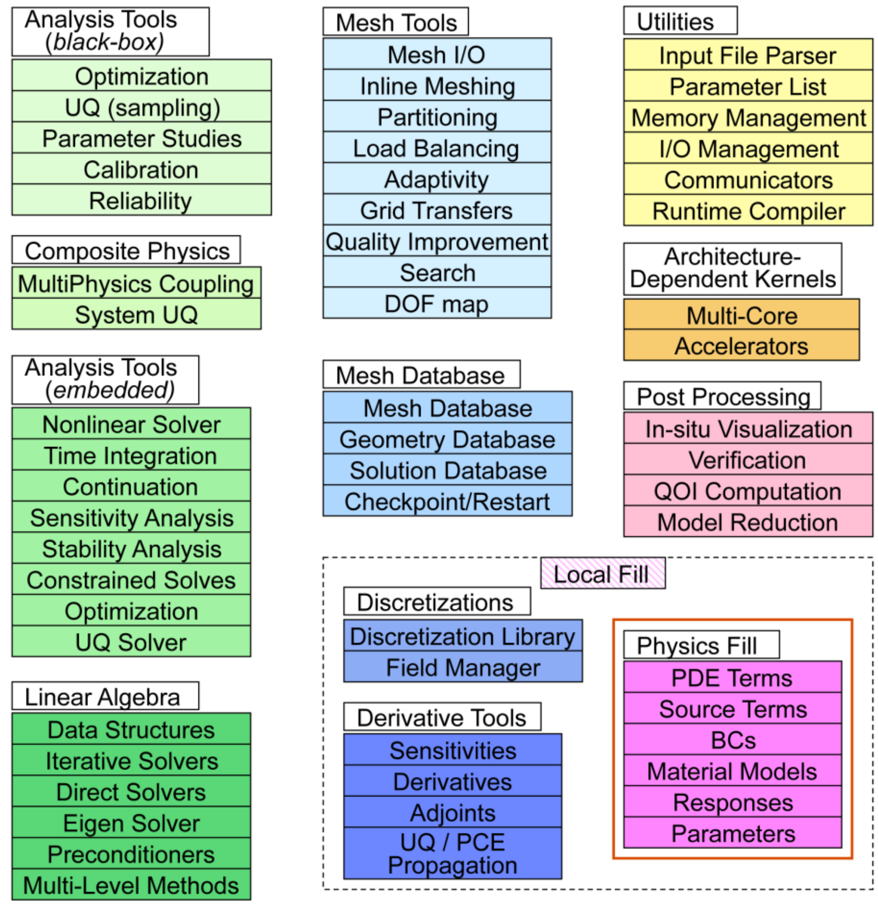
- Focus here is on ***two climate components*** developed at Sandia:
  - Aeras global ***atmosphere*** model.
  - FELIX ***land-ice*** model.

Implemented within the code Albany

Albany Multiphysics Code

Sandia open-source* parallel, C++, multi-physics finite element code.

- ***Component-based*** design for rapid development of new physics & capabilities.
- Extensive use of libraries from the open-source ***Trilinos*** project:
  - Automatic differentiation.
  - Discretizations/meshes, mesh adaptivity.
  - Solvers, time-integration schemes.
  - Performance-portable kernels.
- ***Advanced analysis*** capabilities:
  - Parameter estimation.
  - Uncertainty quantification (DAKOTA).
  - Optimization.
  - Sensitivity analysis.



40+ packages; 120+ libraries

Albany Multiphysics Code

Sandia National Laboratories

Trilinos

Sandia open-source* parallel, C++, multi-physics finite element code.

- ***Component-based*** design for rapid development of new physics & capabilities.
- Extensive use of libraries from the open-source ***Trilinos*** project:
  - Automatic differentiation.
  - Discretizations/meshes, mesh adaptivity.
  - Solvers, time-integration schemes.
  - **Performance-portable kernels.**

This talk

- ***Advanced analysis*** capabilities:
  - Parameter estimation.
  - Uncertainty quantification (DAKOTA).
  - Optimization.
  - Sensitivity analysis.

Analysis Tools (*black-box*)
- Optimization
- UQ (sampling)
- Parameter Studies
- Calibration
- Reliability

Composite Physics
- MultiPhysics Coupling
- System UQ

Analysis Tools (*embedded*)
- Nonlinear Solver
- Time Integration
- Continuation
- Sensitivity Analysis
- Stability Analysis
- Constrained Solves
- Optimization
- UQ Solver

Linear Algebra
- Data Structures
- Iterative Solvers
- Direct Solvers
- Eigen Solver
- Preconditioners
- Multi-Level Methods

Mesh Tools
- Mesh I/O
- Inline Meshing
- Partitioning
- Load Balancing
- Adaptivity
- Grid Transfers
- Quality Improvement
- Search
- DOF map

Mesh Database
- Mesh Database
- Geometry Database
- Solution Database
- Checkpoint/Restart

Utilities
- Input File Parser
- Parameter List
- Memory Management
- I/O Management
- Communicators
- Runtime Compiler

Architecture-Dependent Kernels
- Multi-Core
- Accelerators

Post Processing
- In-situ Visualization
- Verification
- QOI Computation
- Model Reduction

Local Fill

Discretizations
- Discretization Library
- Field Manager

Derivative Tools
- Sensitivities
- Derivatives
- Adjoints
- UQ / PCE Propagation

Physics Fill
- PDE Terms
- Source Terms
- BCs
- Material Models
- Responses
- Parameters

40+ packages; 120+ libraries

* https://github.com/gahansen/Albany.

Performance-portability via *Kokkos*

We need to be able to run climate models on ***new architecture machines*** (hybrid systems) and ***manycore devices*** (multi-core CPU, NVIDIA GPU, Intel Xeon Phi, etc.) .

- In Albany, we achieve performance-portability via ***Kokkos.***
  - ***Kokkos:*** C++ library and programming model that provides performance portability across multiple computing architectures.
    → *Examples:* Multicore CPU, NVIDIA GPU, Intel Xeon Phi, and more.
  - Provides ***automatic access*** to OpenMP, CUDA, Pthreads, etc.
  - Designed to work with the ***MPI+X*** programming model.
  - Abstracts ***data layouts*** for optimal performance ("array of strucs" vs. struct of arrays", locality).

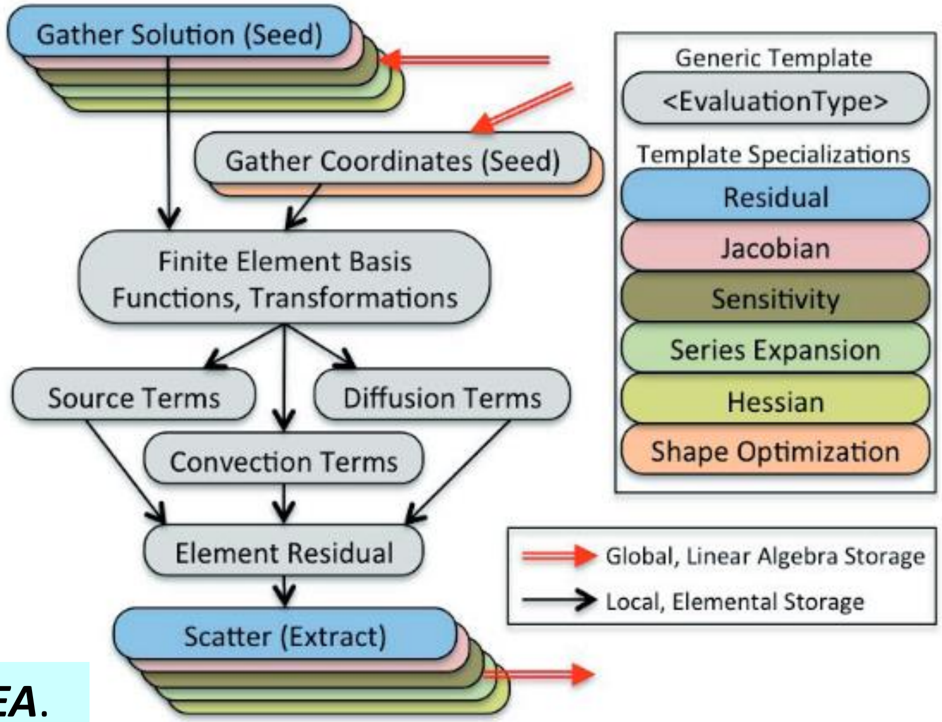
With *Kokkos*, you write an algorithm **once**, and just change a template parameter to get the optimal data layout for your hardware.

→ Allows researcher to focus on ***algorithm development*** for large heterogeneous architectures.

Albany Finite Element Assembly (FEA)

- ***Gather operation*** extracts solution values out of global solution vector.
- Physics ***evaluator*** functions operate on ***workset*** of elements, store evaluated quantities in local field arrays.
- FEA relies on ***template based generic programming*** + ***automatic differentiation*** for Jacobians, tangents, etc.
- ***Scatter operation*** adds local residual, Jacobian to global residual, Jacobian.

Albany performance-portability: focus on ***FEA***.



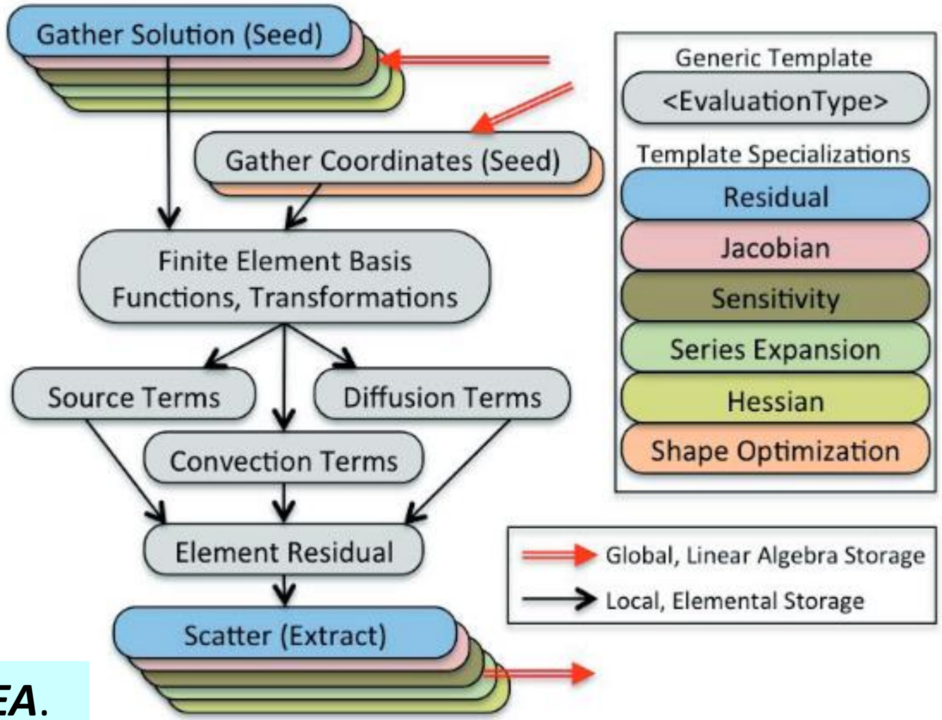
Problem Type	% CPU time for FEA
Implicit	50%
Explicit	99%

Albany Finite Element Assembly (FEA)

- ***Gather operation*** extracts solution values out of global solution vector.
- Physics ***evaluator*** functions operate on ***workset*** of elements, store evaluated quantities in local field arrays.
- FEA relies on ***template based generic programming*** + ***automatic differentiation*** for Jacobians, tangents, etc.
- ***Scatter operation*** adds local residual, Jacobian to global residual, Jacobian.

Albany performance-portability: focus on ***FEA***.

- ***MPI-only FEA:***
  - Each MPI process has workset of cells & computes nested parallel for loops.

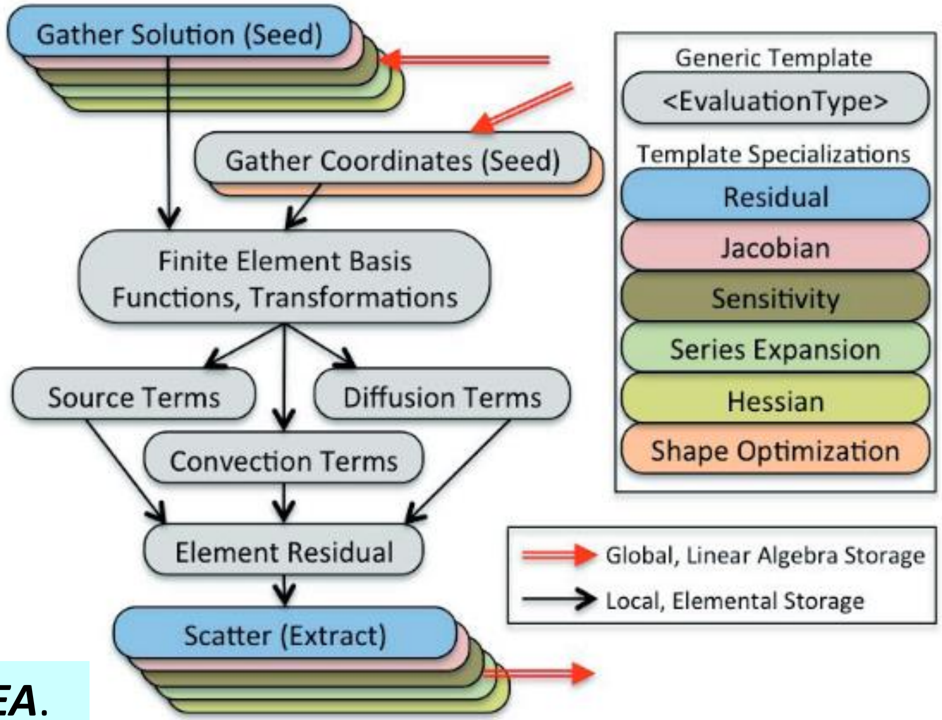


Problem Type	% CPU time for FEA
Implicit	50%
Explicit	99%

Albany Finite Element Assembly (FEA)

Sandia National Laboratories

- ***Gather operation*** extracts solution values out of global solution vector.
- Physics ***evaluator*** functions operate on ***workset*** of elements, store evaluated quantities in local field arrays.
- FEA relies on ***template based generic programming*** + ***automatic differentiation*** for Jacobians, tangents, etc.
- ***Scatter operation*** adds local residual, Jacobian to global residual, Jacobian.



Albany performance-portability: focus on ***FEA***.

- ***MPI-only FEA:***
  - Each MPI process has workset of cells & computes nested parallel for loops.
- ***MPI+X FEA:***
  - Each MPI process has workset of cells.
  - Multi-dimensional parallelism with +X (X=OpenMP, CUDA) for nested parallel for loops.

Problem Type	% CPU time for FEA
Implicit	50%
Explicit	99%

MPI+X FEA via *Kokkos*

- ***MPI-only*** nested for loop:

```
for (int cell=0; cell<numCells; ++cell)
  for (int node=0; node<numNodes; ++node)
    for (int qp=0; qp<numQPs; ++qp)
      compute A;
```

MPI process *n*

MPI+X FEA via *Kokkos*

- ***Multi-dimensional parallelism*** for nested for loops via *Kokkos:*

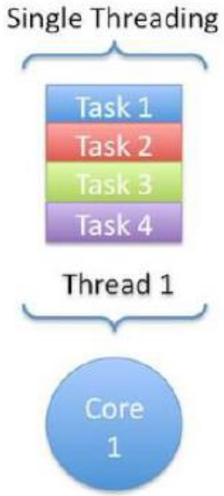
```
for (int cell=0; cell<numCells; ++cell)
   for (int node=0; node<numNodes; ++node)
      for (int qp=0; qp<numQPs; ++qp)
         compute A;                MPI process n
```

Thread *1* computes A for (cell,node,qp)=(0,0,0)

Thread *2* computes A for (cell,node,qp)=(0,0,1)

⋮

Thread *N* computes A for (cell,node,qp)=(numCells,numNodes,numQPs)



MPI+X FEA via *Kokkos*

- ***Multi-dimensional parallelism*** for nested for loops via *Kokkos:*

```
for (int cell=0; cell<numCells; ++cell)
  for (int node=0; node<numNodes; ++node)
    for (int qp=0; qp<numQPs; ++qp)
      compute A;
```

MPI process *n*

Thread *1* computes A for (cell,node,qp)=(0,0,0)

Thread *2* computes A for (cell,node,qp)=(0,0,1)

⋮

Thread *N* computes A for (cell,node,qp)=(numCells,numNodes,numQPs)

```
computeA_Policy range({0,0,0},{(int)numCells,(int)numNodes,(int)numQPs}, computeA_TileSize);
Kokkos::Experimental::md_parallel_for<ExecutionSpace>(range,*this);
```

MPI+X FEA via *Kokkos*

- ***Multi-dimensional parallelism*** for nested for loops via *Kokkos:*

```
for (int cell=0; cell<numCells; ++cell)
  for (int node=0; node<numNodes; ++node)
    for (int qp=0; qp<numQPs; ++qp)
      compute A;                MPI process n
```

Thread *1* computes A for (cell,node,qp)=(0,0,0)

Thread *2* computes A for (cell,node,qp)=(0,0,1)

⋮

Thread *N* computes A for (cell,node,qp)=(numCells,numNodes,numQPs)

```
computeA_Policy range({0,0,0},{(int)numCells,(int)numNodes,(int)numQPs}, computeA_TileSize);
Kokkos::Experimental::md_parallel_for<ExecutionSpace>(range,*this);
```

- ExecutionSpace defined at ***compile time***, e.g.

```
typedef Kokkos::OpenMP ExecutionSpace; //MPI+OpenMP
typedef Kokkos::CUDA ExecutionSpace; //MPI+CUDA
typedef Kokkos::Serial ExecutionSpace; //MPI-only
```

MPI+X FEA via *Kokkos*

- ***Multi-dimensional parallelism*** for nested for loops via *Kokkos:*

```
for (int cell=0; cell<numCells; ++cell)
  for (int node=0; node<numNodes; ++node)
    for (int qp=0; qp<numQPs; ++qp)
      compute A;                MPI process n
```

Thread *1* computes A for (cell,node,qp)=(0,0,0)

Thread *2* computes A for (cell,node,qp)=(0,0,1)

⋮

Thread *N* computes A for (cell,node,qp)=(numCells,numNodes,numQPs)

```
computeA_Policy range({0,0,0},{(int)numCells,(int)numNodes,(int)numQPs}, computeA_TileSize);
Kokkos::Experimental::md_parallel_for<ExecutionSpace>(range,*this);
```

- ExecutionSpace defined at ***compile time***, e.g.

```
typedef Kokkos::OpenMP ExecutionSpace; //MPI+OpenMP
typedef Kokkos::CUDA ExecutionSpace; //MPI+CUDA
typedef Kokkos::Serial ExecutionSpace; //MPI-only
```

- ***Atomics*** used to scatter local data to global data structures

  Kokkos::atomic_fetch_add

MPI+X FEA via *Kokkos*

- ***Multi-dimensional parallelism*** for nested for loops via *Kokkos:*

```
for (int cell=0; cell<numCells; ++cell)
  for (int node=0; node<numNodes; ++node)
    for (int qp=0; qp<numQPs; ++qp)
      compute A;                MPI process n
```

Thread *1* computes A for (cell,node,qp)=(0,0,0)

Thread *2* computes A for (cell,node,qp)=(0,0,1)

⋮

Thread *N* computes A for (cell,node,qp)=(numCells,numNodes,numQPs)

```
computeA_Policy range({0,0,0},{(int)numCells,(int)numNodes,(int)numQPs}, computeA_TileSize);
Kokkos::Experimental::md_parallel_for<ExecutionSpace>(range,*this);
```

- ExecutionSpace defined at ***compile time***, e.g.

  typedef Kokkos::OpenMP ExecutionSpace; //MPI+OpenMP

  typedef Kokkos::CUDA ExecutionSpace; //MPI+CUDA

  typedef Kokkos::Serial ExecutionSpace; //MPI-only

- ***Atomics*** used to scatter local data to global data structures

  Kokkos::atomic_fetch_add

- For MPI+CUDA, data transfer from host to device handled by ***CUDA UVM****.

* Unified Virtual Memory.

Computer Architectures

Performance-portability of FEA in Albany has been tested across ***multiple architectures***: Intel Sandy Bridge, IBM Power8, Keplar/Pascal GPUs, KNL Xeon Phi

- **Shannon** used for testing, performance tests
  - 10 nodes (dual-SandyBridge (16) + K80 dual-GPU)
- **Ride** used for testing, performance tests
  - 12 nodes (dual-Power8 (16) + P100 quad-GPU)
- **Titan** used for full length simulations, performance tests
  - 18,688 nodes (AMD Opteron (16) + K20 GPU)

Atmosphere Model & Project

Energy & Climate

Hydrostatic model
2D (x-z) &
3D

Aeras*
Sandia National Laboratories
Lab-Directed Research and
Development (LDRD)
2014-2016

Albany
SNL
Finite Element
Application Code Base

Shallow water model
(2D)

Increased complexity

Goal: demonstrate next-generation capabilities in an atmosphere model suitable for climate

*Greek for "air"

Next-generation features:

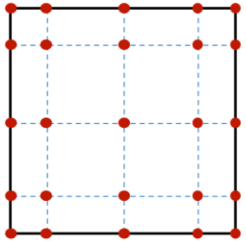
- Performance portability via *Kokkos*.
- Embedded Uncertainty Quantification.

- ***Sandia*** LDRD project involving computational scientists, mathematicians, and climate scientists.
- Led to ***follow-up*** BER project: ***Climate Model Development & Validation*** (CMDV) → key task is Spectral Element Method (SEM) dycore refactore using C++ and ***Kokkos.***

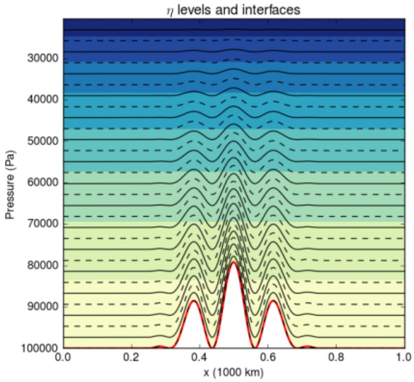
3D Hydrostatic Atmosphere Model

- ***3D hydrostatic equations*** for atmosphere: Navier-Stokes-like model derived under hydrostatic fluid assumptions.

$$
\begin{cases}
\dfrac{\partial \mathbf{u}}{\partial t} + (\zeta + f)\hat{\mathbf{k}} \times \mathbf{u} + \nabla\left(\dfrac{1}{2}\mathbf{u}^2 + \phi\right) + \dot{\eta}\dfrac{\partial \mathbf{u}}{\partial \eta} + \dfrac{RT_v}{p}\nabla p = 0 \\
\phi = \phi_s + \displaystyle\int_{\eta_s}^{\eta} \dfrac{RT}{p}\,\mathrm{d}\eta' \\
\dot{\eta}\dfrac{\partial p}{\partial \eta} = -\dfrac{\partial p}{\partial t} - \displaystyle\int_{\eta_s}^{\eta} \nabla \cdot \dfrac{\partial p}{\partial \eta'}\,\mathrm{d}\eta' \\
RT_v = (c_p - q c_v) T \\
\dfrac{\partial}{\partial t}\dfrac{\partial p}{\partial \eta} + \nabla \cdot \left(\mathbf{u}\dfrac{\partial p}{\partial \eta}\right) + \dfrac{\partial}{\partial \eta}\left(\dot{\eta}\dfrac{\partial p}{\partial \eta}\right) = 0 \\
\dfrac{\partial T}{\partial t} + \mathbf{u} \cdot \nabla T + \dot{\eta}\dfrac{\partial T}{\partial \eta} - \dfrac{RT_v}{c_p p}\omega = 0 \\
\omega = \dfrac{\partial p}{\partial t} + \mathbf{u} \cdot \nabla p
\end{cases}
$$



- Surface of sphere discretized via ***quadrilateral shell spectral*** elements ("cubed sphere" mesh, Gauss-Lobatto quadrature).
- ***Finite difference discretization*** in hybrid vertical coordinate system.
- ***Explicit time-stepping*** via RK methods (diagonal mass).
- Stabilization via ***hyper-viscosity*** $(\tau \nabla^4(\cdot))$



3D Hydrostatic Atmosphere Model

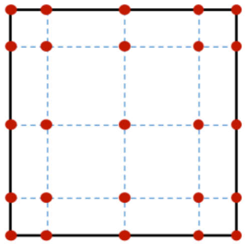
- ***3D hydrostatic equations*** for atmosphere: Navier-Stokes-like model derived under hydrostatic fluid assumptions.

$$
\left\{
\begin{aligned}
&\frac{\partial \mathbf{u}}{\partial t} + (\zeta + f)\hat{\mathbf{k}} \times \mathbf{u} + \nabla\left(\frac{1}{2}\mathbf{u}^2 + \phi\right) + \dot{\eta}\frac{\partial \mathbf{u}}{\partial \eta} + \frac{RT_v}{p}\nabla p = 0\\
&\phi = \phi_s + \int_{\eta_s}^{\eta} \frac{RT}{p} d\eta'\\
&\dot{\eta}\frac{\partial p}{\partial \eta} = -\frac{\partial p}{\partial t} - \int_{\eta_s}^{\eta} \nabla \cdot \frac{\partial p}{\partial \eta'} d\eta'\\
&RT_v = (c_p - q c_v)T\\
&\frac{\partial}{\partial t}\frac{\partial p}{\partial \eta} + \nabla \cdot \left(\mathbf{u}\frac{\partial p}{\partial \eta}\right) + \frac{\partial}{\partial \eta}\left(\dot{\eta}\frac{\partial p}{\partial \eta}\right) = 0\\
&\frac{\partial T}{\partial t} + \mathbf{u}\cdot\nabla T + \dot{\eta}\frac{\partial T}{\partial \eta} - \frac{RT_v}{c_p p}\omega = 0\\
&\omega = \frac{\partial p}{\partial t} + \mathbf{u}\cdot\nabla p
\end{aligned}
\right.
$$

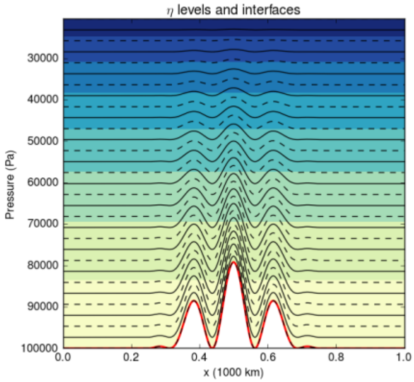
Explicit: FEA is 99% CPU-time

+ X parallelization over ***cells***, ***nodes*** and ***levels***.

Runs performed on ***Ride*** cluster at Sandia.



- Surface of sphere discretized via ***quadrilateral shell spectral*** elements ("cubed sphere" mesh, Gauss-Lobatto quadrature).
- ***Finite difference discretization*** in hybrid vertical coordinate system.
- ***Explicit time-stepping*** via RK methods (diagonal mass).
- Stabilization via ***hyper-viscosity*** ($\tau\nabla^4(\cdot)$)

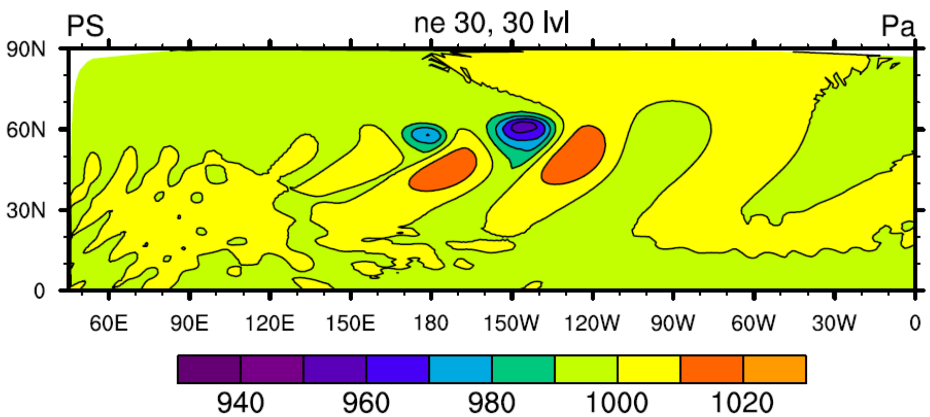


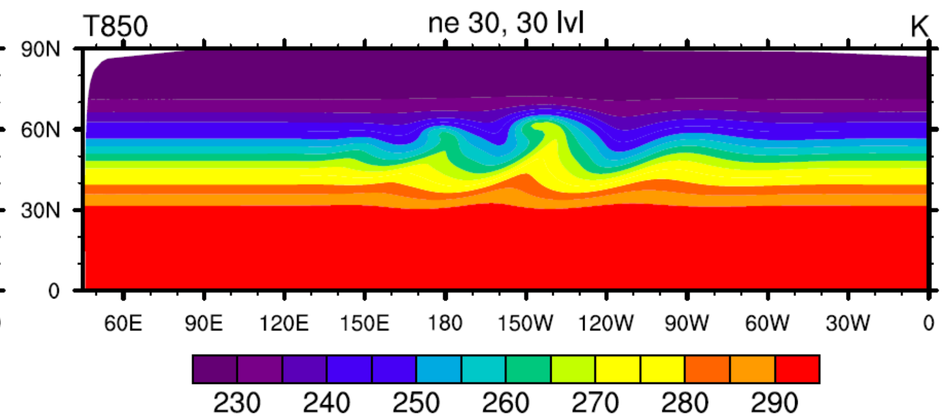
Baroclinic Instability Test Case

- Meshes/parameters considered:

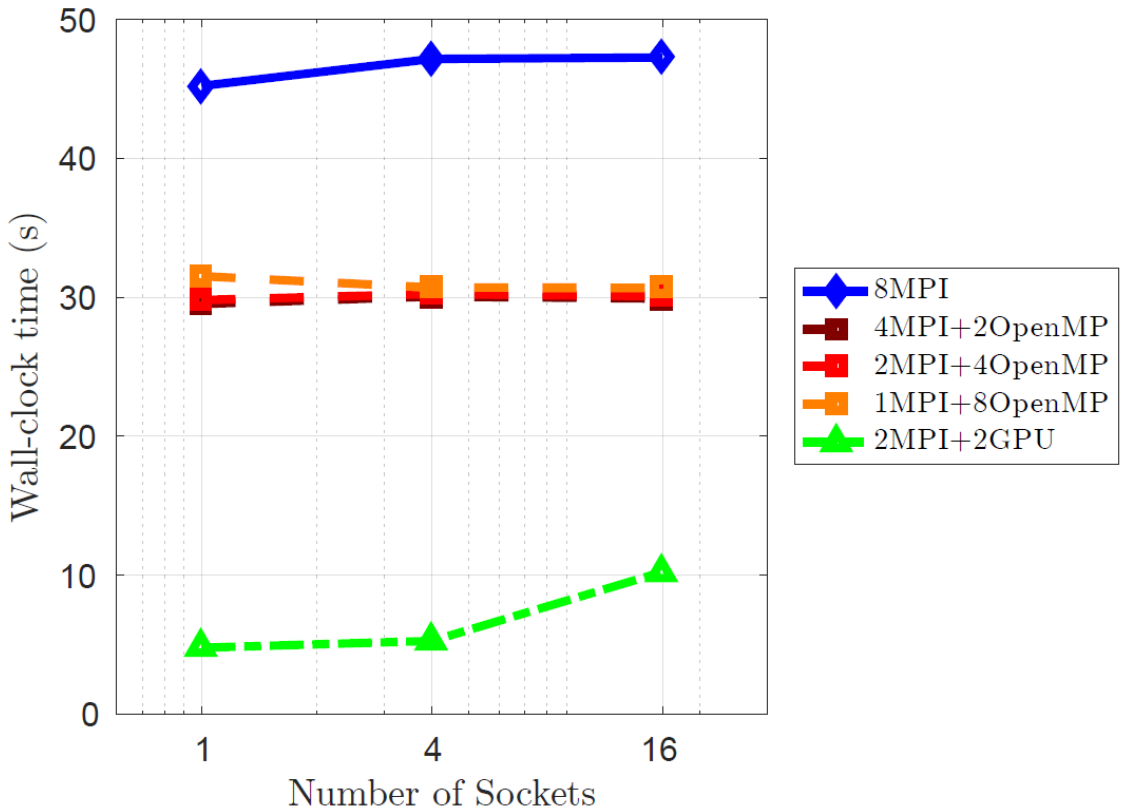
Mesh	Resolution	# Elements	Fixed dt	Hyperviscosity Tau
uniform_30	1°	5,400	30	5.0e15
uniform_60	0.5°	21,600	10	1.09e14
uniform_120	0.25°	86,400	5	1.18e13

- 100 explicit RK4 iterations, 3rd order elements, 10 levels





Weak Scalability (FEA)

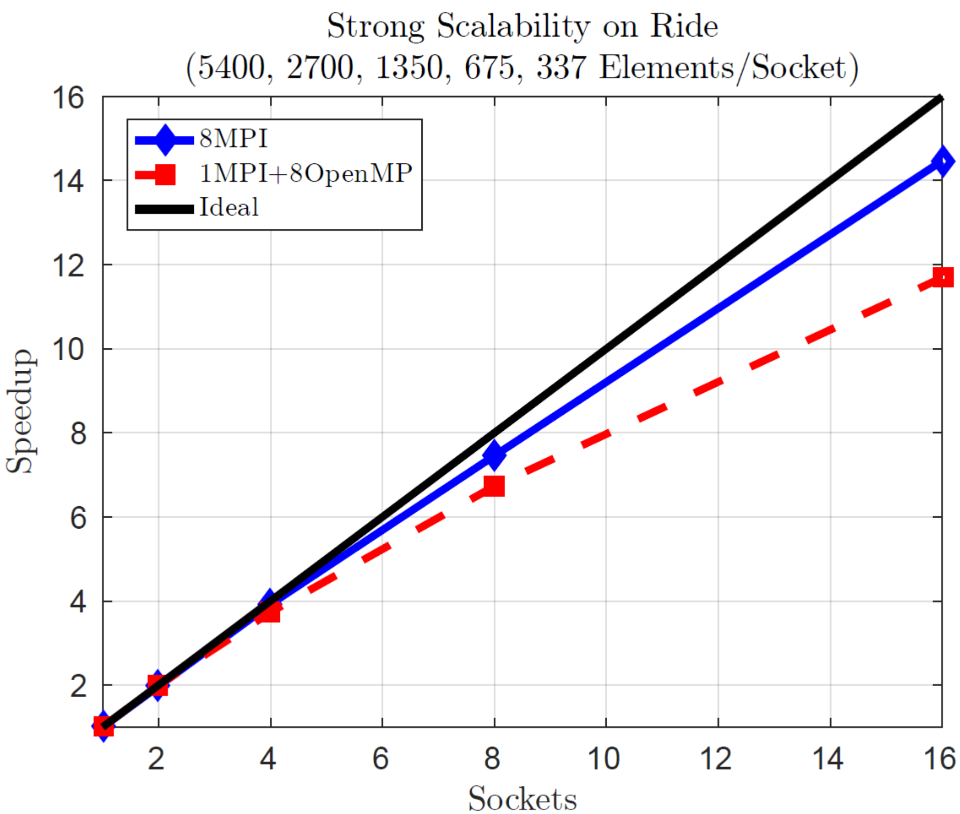


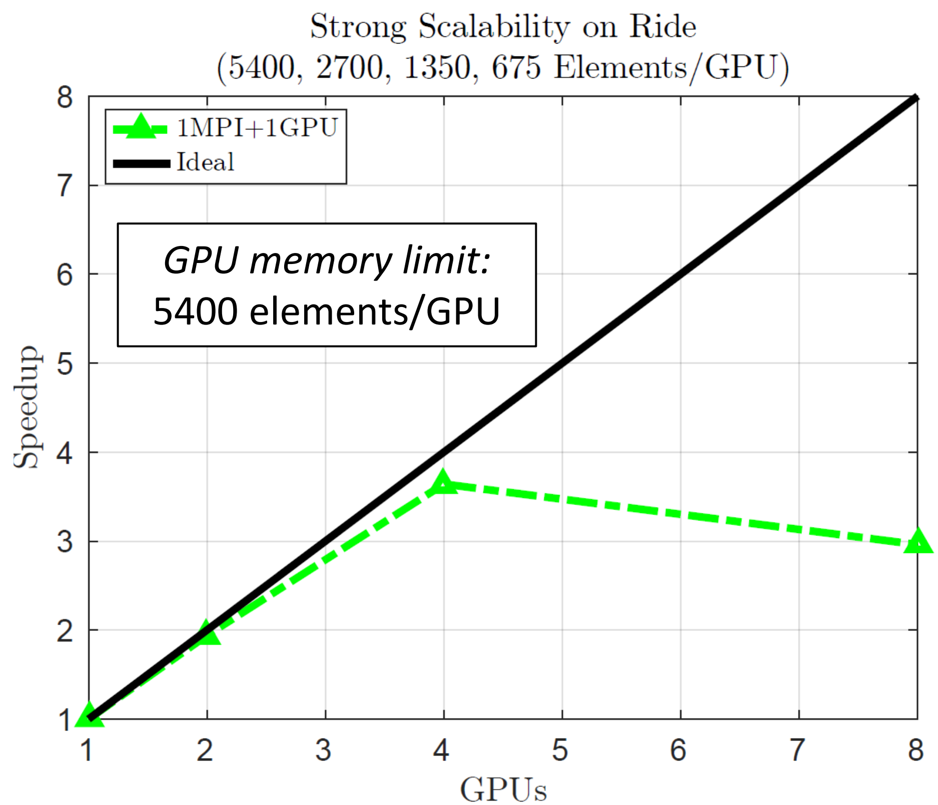
Ride:

- 12 nodes (2 sockets/node)
- 8 Power8 cores/socket
- 2 P100 GPUs/socket
- 16GB/GPU

- 3 mesh resolutions: uniform_30, uniform_60, uniform_90
- Good weak scaling MPI, MPI+OpenMP
- Worse GPU weak scaling: communication bottlenecks between sockets (GPU→ CPU → CPU → GPU data movement many times using CUDA UVM).

Strong Scalability (FEA)





- ~75% efficiency for MPI+OpenMP
- Poor GPU strong scaling for > 4 GPUs: memory transfers dominate.
  → May be improved by replacing CUDA UVM w/ manual memory transfer.

FELIX Land-Ice Solver & PISCEES Project

PISCEES*
SciDaC Application Partnership
(DOE's BER + ASCR divisions)
2012-2017

3 land-ice dycores developed under PISCEES

FSU FELIX
FSU
Finite Element
Full Stokes Model

Albany/FELIX**
SNL
Finite Element
"First Order" Stokes Model

BISICLES
LBNL
Finite Volume
L1L2 Model

Increased fidelity

Goal: support DOE climate missions (sea-level rise predictions)

*Predicting Ice Sheet Climate & Evolution at Extreme Scales.
**Finite Elements for Land Ice eXperiments

- ***Multi-lab/multi-university*** project involving mathematicians, climate scientists, and computer scientists.
- Leverages software/expertise from ***SciDAC Institutes*** (FASTMath, QUEST, SUPER) and hardware from ***DOE Leadership Class Facilities***.

First-Order Stokes Land-Ice Model

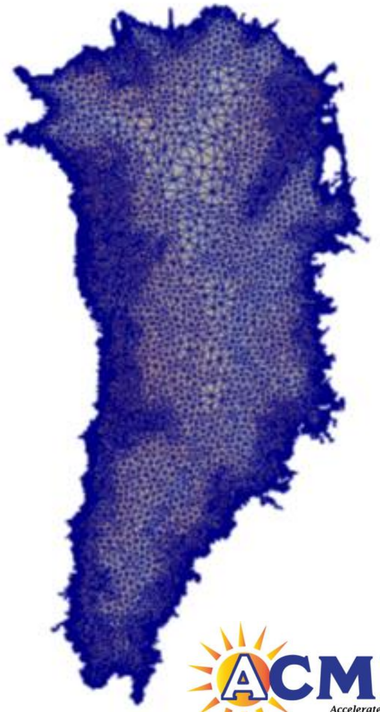
- Ice velocities given by the ***"First-Order" Stokes PDEs*** with nonlinear viscosity:

$$\begin{cases} -\nabla\cdot(2\mu\dot{\epsilon}_1) = -\rho g \dfrac{\partial s}{\partial x} \\ -\nabla\cdot(2\mu\dot{\epsilon}_2) = -\rho g \dfrac{\partial s}{\partial y} \end{cases}$$

$$\mu = \frac{1}{2}A^{-\frac{1}{n}}\left(\frac{1}{2}\sum_{ij}\dot{\epsilon}_{ij}{}^2\right)^{\left(\frac{1}{2n}-\frac{1}{2}\right)}$$

$$\dot{\epsilon}_1{}^T = (2\dot{\epsilon}_{11}+\dot{\epsilon}_{22}, \dot{\epsilon}_{12}, \dot{\epsilon}_{13})$$
$$\dot{\epsilon}_2{}^T = (2\dot{\epsilon}_{12}, \dot{\epsilon}_{11}+2\dot{\epsilon}_{22}, \dot{\epsilon}_{23})$$
$$\dot{\epsilon}_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j}+\frac{\partial u_j}{\partial x_i}\right)$$

- ***3D unstructured grid*** FEM discretization.
- ***Newton method*** nonlinear solver with automatic differentiaton Jacobians.
- ***Algebraic-multigrid**** preconditioned Krylov linear solvers.
- ***Advanced analysis capabilities***: deterministic inversion, calibration, UQ.
- As part of ***ACME DOE ESM***, FELIX will be used to provide actionable predictions of 21st century sea-level rise.



* Previous talk by Ray Tuminaro.

First-Order Stokes Land-Ice Model

- Ice velocities given by the ***"First-Order" Stokes PDEs*** with nonlinear viscosity:

$$\begin{cases} -\nabla \cdot (2\mu \dot{\epsilon}_1) = -\rho g \dfrac{\partial s}{\partial x} \\ -\nabla \cdot (2\mu \dot{\epsilon}_2) = -\rho g \dfrac{\partial s}{\partial y} \end{cases}$$

$$\mu = \frac{1}{2} A^{-\frac{1}{n}} \left(\frac{1}{2} \sum_{ij} \dot{\epsilon}_{ij}^{\,2} \right)^{\left(\frac{1}{2n} - \frac{1}{2}\right)}$$

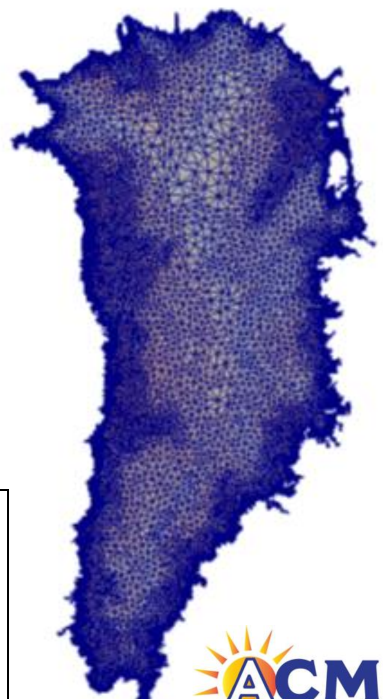
$$\dot{\epsilon}_1^{\,T} = (2\dot{\epsilon}_{11} + \dot{\epsilon}_{22}, \dot{\epsilon}_{12}, \dot{\epsilon}_{13})$$
$$\dot{\epsilon}_2^{\,T} = (2\dot{\epsilon}_{12}, \dot{\epsilon}_{11} + 2\dot{\epsilon}_{22}, \dot{\epsilon}_{23})$$
$$\dot{\epsilon}_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right)$$

- ***3D unstructured grid*** FEM discretization.
- ***Newton method*** nonlinear solver with automatic differentiaton Jacobians.
- ***Algebraic-multigrid**** preconditioned Krylov linear solvers.
- ***Advanced analysis capabilities***: deterministic inversion, calibration, UQ.
- As part of ***ACME DOE ESM***, FELIX will be used to provide actionable predictions of 21st century sea-level rise.

Implicit: FEA is 50% CPU-time

+ X parallelization over ***cells*** only.

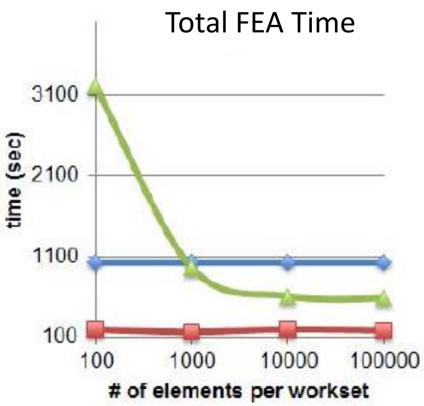
Runs performed on ***Shannon*** cluster at Sandia and ***Titan*** supercomputer



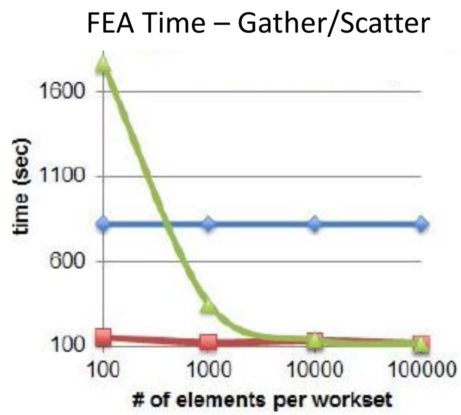
4km Greenland & 8km Antarctica on *Shannon*

Sandia National Laboratories

4km Greenland

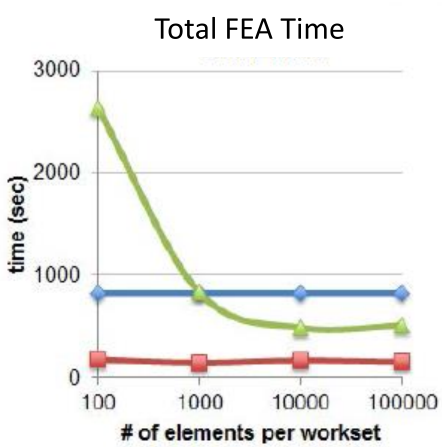


(a)

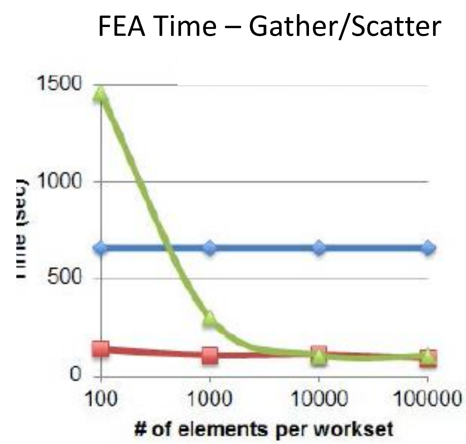


(b)

8km Antarctica



(c)



(d)

Shannon: *32 nodes*

- *2 8-core Sandy Bridge Xeon E5-2670 @ 2.6GHz (HT deactivated)/node.*
- *128GB DDR3 memory/node*
- *2x NVIDIA K20x/node.*

— Serial: 1 MPI thread/node
— OpenMP: 16 OpenMP threads/node
— CUDA: 1 GPU/node

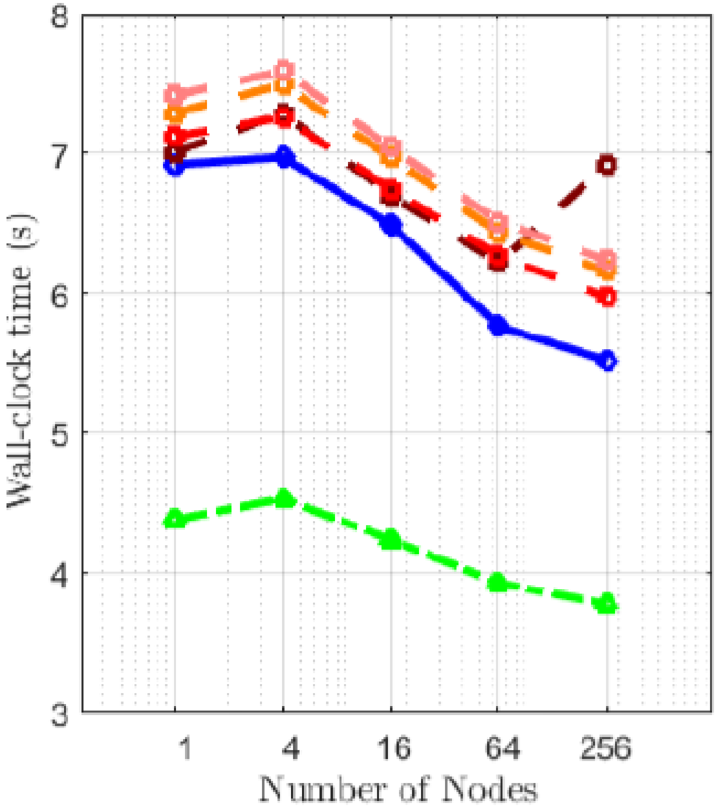
Max speedup over Serial for workset size > 1000		
	OpenMP	CUDA
Total FEA Time	5.6x	1.7x
FEA Time – Gather/Scatter	7.2x	6.7x

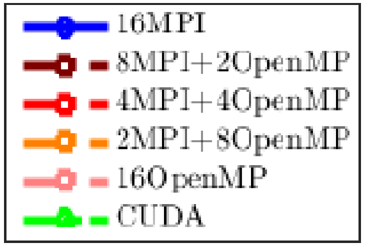
"# of elements/workset" = threading index (allows for on-node parallelism)

Greenland Weak Scalability on *Titan*

Weak scalability on *Titan* (16km, 8km, 4km, 2km, 1km Greenland)

Wall-clock Time: FEA

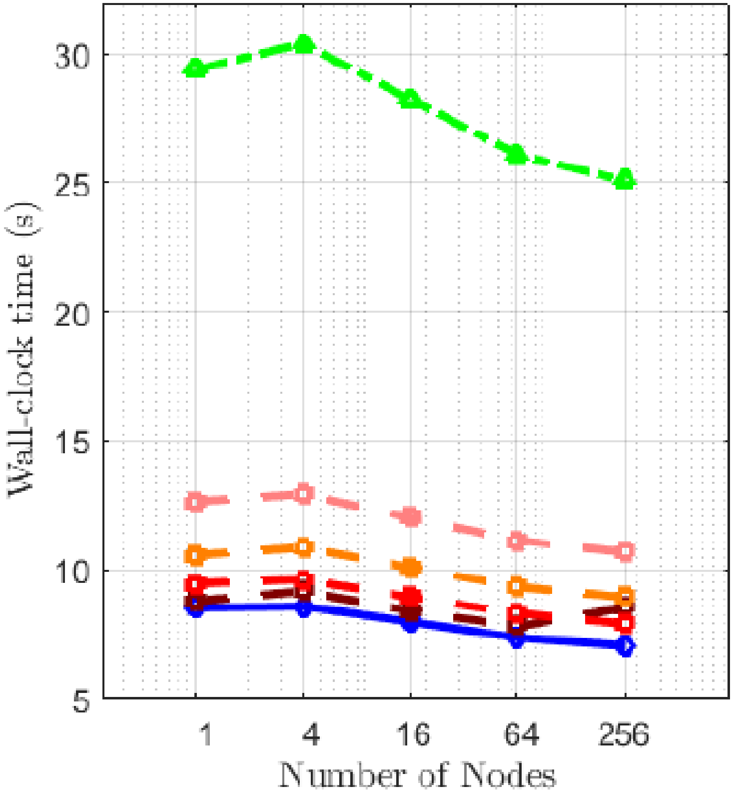




Titan: 18,688 AMD Opteron nodes

- 16 cores per node
- 1 K20X Kepler GPUs/ node
- 32GB + 6GB memory/ node

Wall-clock Time:
Total Time – Setup Time



Summary/Conclusions

- A ***performance portable*** implementation of the ***Aeras next-generation global atmosphere*** model and ***FELIX land-ice*** model was created using *Kokkos* within the Albany code base.

- With this implementation, the ***same code*** can run on devices with drastically ***different memory models*** (many-core CPU, NVIDIA GPU, Intel Xeon Phi, etc.).

- ***Heterogeneous HPC architectures*** can now be utilized for climate research in Aeras and FELIX.

- Performance studies show that ***further optimization*** is needed to fully utilize all resources.

More on ***performance-portability*** of Albany using *Kokkos* can be found here: https://github.com/gahansen/Albany/wiki/Albany-performance-on-next-generation-platforms

Ongoing/Future Work

- ***Profiling*** using TAU and nvprof.
- Methods for ***improving performance***:
  - Reduce excess memory usage.
  - Utilize shared memory.
  - Replace CUDA UVM with manual memory transfer.
  - Improve performance of other sections of code besides FEA.
  - Parallelize over nodes and quadrature points in addition to cells for FELIX.

- Performance-portability of ***preconditioned iterative linear solve*** using *Kokkos* for implicit problems in Albany (e.g., FELIX).
- ***Journal article*** on this work in preparation:

I. Demeshko, W. Spotz, J. Watkins, **I. Tezaur**, O. Guba, A. Salinger, R. Pawlowski, M. Heroux. "Towards performance-portability of the Albany finite element analysis code using the Kokkos library", *J. HPC Appl.* (in preparation).

Appendix: Parallelism on Modern Hardware

Year	Memory Access Time	Single Core Cycle Time
1980s	~100 ns	~100 ns
Today	~50-100 ns	~1 ns

- ***Memory access time*** has remained the ***same***.
- ***Single core*** performance has ***improved*** but ***stagnated***.
- ***Computations*** are ***cheap***, ***memory transfer*** is ***expensive***.
- ***More performance*** from ***multicore/manycore*** processors.

Appendix: *Kokkos*-ification of Finite Element Assembly (Example)

```
typedef Kokkos::OpenMP ExecutionSpace;
//typedef Kokkos::CUDA ExecutionSpace;
//typedef Kokkos::Serial ExecutionSpace;
template<typename ScalarT>
vectorGrad<ScalarT>::vectorGrad()
{
Kokkos::View<ScalarT****, ExecutionSpace> vecGrad("vecGrad", numCells, numQP, numVec, numDim);
}
**********************************************
template<typename ScalarT>
void vectorGrad<ScalarT>::evaluateFields()
{
  Kokkos::parallel_for<ExecutionSpace> (numCells, *this);
}
**********************************************
template<typename ScalarT>
KOKKOS_INLINE_FUNCTION
void vectorGrad<ScalarT>:: operator() (const int cell) const
{
  for (int cell = 0; cell < numCells; cell++)
  for (int qp = 0; qp < numQP; qp++) {
    for (int dim = 0; dim < numVec; dim++) {
      for (int i = 0; i < numDim; i++) {
        for (int nd = 0; nd < numNode; nd++) {
          vecGrad(cell, qp, dim, i) += val(cell, nd, dim) * basisGrad(nd, qp, i);
} } } } }
```

ExecutionSpace parameter tailors code for device (e.g., OpenMP, CUDA, etc.)

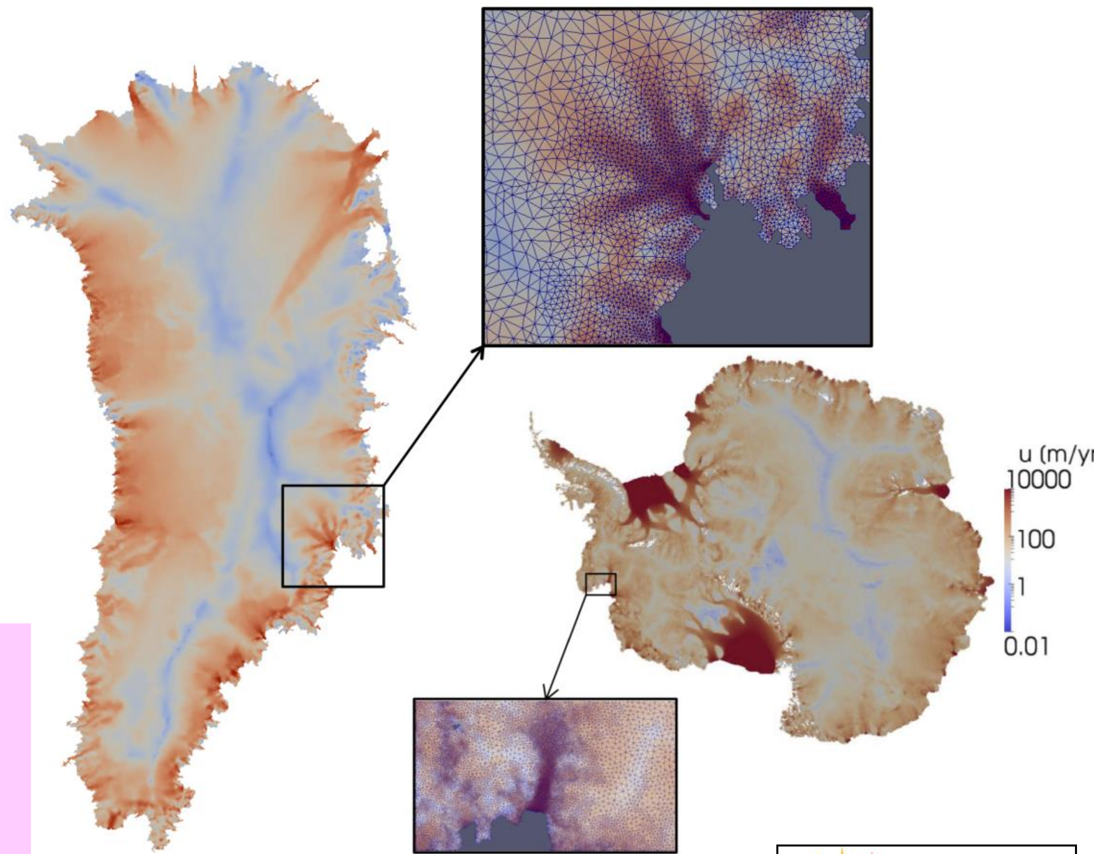
Appendix: PISCEES Land-Ice Project

Sandia's Role in the PISCEES Project: to **develop** and **support** a robust and scalable land ice solver based on the "First-Order" (FO) Stokes equations → *Albany/FELIX**

Requirements for *Albany/FELIX*:

- ***Unstructured grid*** finite elements.
- ***Verified, scalable, fast, robust***
- ***Portable*** to new/emerging architecture machines (multi-core, many-core, GPU)
- ***Advanced analysis*** capabilities: deterministic inversion, calibration, uncertainty quantification.

As part of **ACME *DOE earth system model***, solver will provide actionable predictions of 21st century sea-level rise (including uncertainty).



*Finite Elements for Land Ice eXperiments